

24

FORTH MUMPS LPB etc...

MAI 1986

20 Fr

F 83

JEDI AIR FORT

EDITORIAL

Deux ans déjà ! JEDI fête ce mois-ci son second anniversaire. Ceci n'aurait pas été possible s'il n'existait pas un réel intérêt parmi vous pour une programmation structurée et l'emploi de langages autres que le BASIC. De plus, il devenait indispensable de créer une revue qui traite d'autres choses que de "casses-briques" ou de bancs d'essais de micros. C'est de cette volonté que JEDI est né. Depuis, il a évolué vers la création d'outils logiciels suffisamment portables pour être utilisables par chacun d'entre vous.

Ainsi en est-il à l'heure actuelle. Ainsi en sera-t-il de plus en plus dans le futur. JEDI n'est pas qu'une simple revue. Son principe de diffusion répond à des impératifs très précis. Ne concerner que les personnes réellement intéressées par l'informatique (sus aux badauds !!) et rester to-ta-le-ment indépendants. Deux ans d'existence sans autre soutien financier que le montant des adhésions, qui peut se targuer d'une telle performance ?

Mais JEDI n'est devenu ceci que grâce à vous tous, à vous qui le faites connaître, à vous qui nous envoyez vos réalisations, à vous qui l'appréciez et qui savez nous le dire.

Encore une fois, et au nom de toute l'équipe de rédaction, un grand merci !!!

SOMMAIRE

ASSOCIATION JEDI 8, rue Poirier de Narçay 75014 PARIS

Tel: (1) 45.42.88.90 (de 10h à 18h)

Christian SCHERER

Un compilateur musical portable en LPB

1. Varions un peu !

Notre précédente série consacrée au L.P.B. (Langage Pseudo-Basic) a pu donner à penser à d'aucuns que ce langage était plus spécialement destiné aux applications graphiques.

En fait il n'en est rien, et, pour pour bien montrer la vocation universelle de cet outil, nous traiterons maintenant de la composition musicale, en choisissant le matériel THOMSON comme support de notre exemple car il s'agit d'un matériel maintenant largement répandu dans notre pays.

On objectera peut-être :

- que la composition musicale ne nécessite pas des traitements aussi complexes et surtout aussi rapides que l'animation d'image.
- qu'il existe déjà, dans le BASIC MICROSOFT livré avec ces machines, une instruction PLAY qui réalise déjà toutes les fonctions musicales que nous allons compiler.
- que les posibilités musicales offertes par le constructeur national ne sont peut-être pas des plus extraordinaires.

Nous braverons par avance toutes ces objections, avec la foi et la certitude que nous donnent nos propres arguments :

- nous voulons justement montrer que le langage L.P.B. permet d'éviter, après compilation, de s'affranchir totalement du BASIC.
- nous voulons illustrer la structure du L.P.B. par un exemple facile à comprendre par ceux qui ne sont pas encore familiers avec ce langage.
- partant de cet exemple, on pourra ensuite plus facilement migrer le compilateur musical vers des machines telles que COMMODORE et AMSTRAD, qui ne disposent pas naturellement des bonnes primitives, alors que leurs possibilités techniques sont excellentes.
- enfin, nous sommes fiers de montrer le dernier né de la famille des compilateurs LPB, dans lequel on a introduit de nouvelles notions telles que les procédures et les passages de paramètres que l'on ne trouve que dans des langages très évolués (L.S.E., SUPERBASIC et la famille PASCAL).

2. Un programe versatile.

Sans doute à l'intention des programmeurs astucieux, le constructeur national, déjà cité, a su ménager entre les familles MO5 et TO7 (nous passerons ici sous silence le cas du TO9) de subtiles différences. Pour pouvoir s'adpater automatiquemment à l'une et l'autre des familles, nos programmes doivent comporter des bascules internes (on dit en anglais des "switches"). Ces bascules peuvent être manoeuvrées manuellement, soit au moment de la compilation, soit au moment de l'exécution. Dans le meilleur des cas, un logiciel "bien élevé" sera capable de reconnaître tout seul la machine sur

→

laquelle il est en train de fonctionner, plutôt que de le demander stupidement à l'utilisateur.

Les machines THOMSON possèdent un programme résident, appelé "moniteur", qui joue un rôle analogue au "kernal" des COMMODORE, ou au BIOS (Basic Input-Output System) des machines plus évoluées.

Ce moniteur contient un ensemble de fonctionnalités indispensables pour gérer commodément les communications avec le clavier, l'écran, le générateur de sons, et les autres périphériques.

Sur TO7, on accède à ces "primitives" à travers un vecteur de branchement situé à l'adresse hexadécimale E800. Sur MO5, le même résultat est obtenu grâce à l'instruction-machine SWI suivie d'un octet caractéristique de la primitive demandée. Notre exemple LPB montre comment reconstituer sur un MO5 un vecteur de SWI qui simule le cas du TO7.

De même, pour varier, les deux machines n'implantent pas aux mêmes adresses les variables de travail du moniteur, et notre programme devra donc en tenir compte. Voici la table de correspondance :

MO5	TO7	Variable	Signification
2039	6031	TEMPO	Tempo des séquences musicales
203B	6033	DUREE	Durée des notes
203E	6036	OCTAVE	Sélection d'octave
203E	6035	TIMBRE	Sélection de timbre

3. Choix d'un langage musical.

Ayant reconnu les caractéristiques de notre machine, et tout en gardant à l'esprit que les autres machines-cibles (COMMODORE, AMSTRAD, MSX, SPECTRUM, IBM-PC, etc) ont des caractéristiques différentes, il nous faut définir un langage-source musical aussi universel que possible.

Parmi les machines étudiées, quelques-unes seulement possèdent d'origine un tel langage musical évolué, sous forme d'une extension du BASIC MICROSOFT livré avec la machine. Nous écarterons une première solution, qui utilise sur MSX et IBM-PC la notation musicale anglo-saxonne (A, B, C, ...) pour lui préférer la seconde, propre au matériel THOMSON, plus proche de nos habitudes nationales : DO, RE, MI ...etc ...

Nous ne commettrons pas l'erreur d'inventer un troisième langage musical, alors que notre hexagone est déjà bien trop petit pour rentabiliser à lui seul des développements logiciels lourds.

Dès lors, la syntaxe source est fixée et nous renvoie à la documentation d'accompagnement des matériels THOMSON :

Une note est caractérisée par :

- sa position dans la gamme (DO RE MI FA SO LA SI)
- une altération éventuelle : (b pour un bémol, # pour un dièse)
- une durée relative : L12 pour une noire, L24 pour une blanche, etc

→

- un timbre : A0 pour une onde sinusoïdale, A1, A2 ... pour des altérations différentes

S'y ajoutent :
- la pause : P
- le tempo général : T1, T2 ... T255 : pour l'ensemble des notes

Avec une telle notation, le classique "Au clair de la lune" devient aussi clair que :

```
DATA  A0T30L6SOSOSOLAL12SILAL6SOSILALAL24SO
DATA       L6SOSOSOLAL12SILAL6SOSILALAL24SO
DATA       L6LALALALAL12MIMIL6LASOFA MIL24RE
DATA       L6SOSOSOLAL12SILAL6SOSILALAL24SO
DATA  0
```

4. L'interpréteur musical.

La conversion entre le langage musical source et le langage interne du moniteur s'effectue par l'intermédiaire d'un petit automate de reconnaissance syntaxique écrit en LPB, directement inspiré de la solution MICROSOFT, qui ne présente pas de difficulté particulière.

Dans le cas du langage LPB sur THOMSON, on dispose maintenant de la notion de procédure, avec passage de paramètres (par valeur exclusivement), qui permet une programmation mieux structurée et donc plus facile à lire.

La transposition aux autres micro-ordinateurs ne soulève pas de difficulté insurmontable, dans les limites bien sûr, des possibilités d'expression musicale propres à chacun d'entre eux.

Nous vous invitons à nous faire part de vos expériences propres par l'intermédiaire du journal, qui transmettra, et sommes prêts à publier, si nécessaire, les variantes adaptées aux autres machines citées.

```
200 REM ================ INTERPRETEUR MUSICAL/SOURCE LPB =====
201 :
202 'DEFB BASE SYN &HA000,D SYN &HDF00                :REM TO7 NRDOS
204  DEFB BASE SYN &H6000,D SYN &H5F00                :REM MO5 NRDOS
207 ORG BASE
210 REM ================ CARACTERISTIQUES TO7 ================
211 :
212 'DEFW USERAF SYN &H602D, TEMPO SYN &H6031, DUREE SYN &H6033, OCTAVE SYN &H6036
213 'DEFB TIMBRE SYN &H6035, FORME SYN &H6038, PIA SYN &HE7C3, ECRAN SYN &H4000, ATT SYN &H4C
219 :
220 REM ================ CARACTERISTIQUES MO5 ================
221 :
222  DEFW USERAF SYN &H2070, TEMPO SYN &H2039, DUREE SYN &H203B, OCTAVE SYN &H203E
223  DEFB TIMBRE SYN &H203E, FORME SYN &H2029, PIA SYN &HA7C0, ECRAN SYN &H0, ATT SYN &H70
229 :
230 REM =====================================================
231 :
232 GOTO MAIN
233 :
241 'TEMPO  =   1..255
242 'DUREE  =   1..96
243 'TIMBRE =   0..255
244 'OCTAVE =   16,8,4,2,1
245 'FORME  =   -8..+15   Couleur des affichages graphiques
251 :
363 CHUNK DEFB BYTE
364 CHUNK DEFW WORD
365 INSPECT U
369 'DEFB H SYN &HE800        : REM TO7  Moniteur
370  DEFB H(0)                : REM MO5  Moniteur
371   SWI:DATA &H80,0
372   SWI:DATA &H82,0 :REM PUTC$ = H(3)
373   SWI:DATA &H8A,0 :REM GETC$ = H(6)
374   SWI:DATA &H8C,0 :REM KTST$ = H(9)
375   SWI:DATA &H8E,0 :REM DRAW$ = H(12)
376   SWI:DATA &H90,0 :REM PLOT$ = H(15)
377   SWI:DATA &HA4,0 :REM RSCO$ = H(18)
378   SWI:DATA &HA0,0 :REM K7CO$ = H(21)
379   SWI:DATA &H98,0 :REM GETL$ = H(24)
380   SWI:DATA &H96,0 :REM LPIN$ = H(27)
381   SWI:DATA &H9E,0 :REM NOTE$ = H(30)
382   SWI:DATA &H94,0 :REM GETP$ = H(33)
383   SWI:DATA &H9A,0 :REM GETS$ = H(36)
384   SWI:DATA &H9C,0 :REM JOYS$ = H(39)
425 :
430 PROCEDURE SOUTCHAR(B)
435  GOTO H(3)
449 :
```

```
699 :
700 PROCEDURE SNOTE(B)
709  GOTO H(30)
710 :
1300 REM ============== PLAY =================
1301 :
1302 CHUNK DEFB FILL,LIMLOW,LIMHIGH;DEFW ADDRESS
1303 DEFB TEMPOLSB   SYN TEMPO/LSB
1304 DEFB DUREELSB   SYN DUREE/LSB
1305 DEFB OCTAVELSB  SYN OCTAVE/LSB
1316 :
1318 DEFB SOUNDCHARS(0)
1320 DATA "T",1,255,@TEMPOLSB
1321 DATA "A",0,255,@TIMBRE
1322 DATA "L",1, 96,@DUREELSB
1323 DATA "O",1,  5,@OCTAVELSB
1325 :
1330 DEFB NOTES(0)
1331  DATA "OD1ER3IM5AF6OS8AL:IS<"
1335 :
1340 DEFB OCTAVES(0)
1341  DATA 16,8,4,2,1
1345 :
1346 LABEL FCERROR
1347 SOUTCHAR(7):GOTO FCERROR
1348 :
1350 PROCEDURE CHECKDIGIT(A) : REM GIVING FLC
1351  IF A>="0" THEN A=A-&H3A-&HC6
1352 RETURN
1353 :
1360 PROCEDURE SCANINT(Y) GIVING B
1362  CHECKDIGIT((Y+)):IF FLC THEN FCERROR ELSE B=0
1363  LABEL LOOPDIGIT
1364   A=A-"0":PUSH A:A=10
1365   AB=A MUL B:IF A<>0 THEN FCERROR
1366   B=B+(S+):IF FLC THEN FCERROR
1367   CHECKDIGIT((Y+)):IF NOT FLC THEN LOOPDIGIT
1368  Y=Y-1
1369  RETURN
1370 :
1371 PROCEDURE PLAYNOTE(Y)
1372  B=X/2:A=(Y+):IF A="#"THEN B=B+1 ELSE IF A="b" THEN B=B-1 ELSE Y=Y-1
1373  IF B<=&H30 THEN B=&H3C
1374  SNOTE(B)
1375 RETURN
1376 :
1380 PROCEDURE SCANMUS(Y)
1381  A=(Y+)
1382  IF A="P" THEN SNOTE(&H30):RETURN
1383  B=A:A=(Y+):IF = THEN FCERROR
1384  FOR X=@NOTES STEP 3 UNTIL @NOTES(21)
1385   IF AB=(X) THEN PLAYNOTE(Y):RETURN
1386  NEXT X
1387  :
1388  FOR X=@SOUNDCHARS STEP 5 UNTIL @SOUNDCHARS(20)
1389   IF B=(X) THEN 1391
1390  NEXT X:GOTO FCERROR
1391  B=SCANINT(Y-1)
1392  REM IF B<X/LIMLOW OR B>X/LIMHIGH THEN FCERROR
1394  IF X=@SOUNDCHARS(15) THEN U=@OCTAVES(-1):B=U/B
1396  [X/ADDRESS]=B
1397 RETURN
1398 :
1400 PROCEDURE SPLAY(Y)
1401  A=(Y):IF<>THEN SCANMUS(Y):GOTO SPLAY
1403 :
7090 LABEL MARJOLAINE
7091 DATA "A0T15O4L12DOL6DOL12SOL6SOL12FAL6SOL18MIb"
7092 DATA "L12MIbL6REDOREMIbL12REL6DOL12O3SIL6SO"
7093 DATA "O4L12DOL6DOL12SOL6SOL12FAL6SOL18MIb"
7094 DATA "O4L18FASOL6REMIbREL18DO",0
7099 :
8000 LABEL MAIN
8010  SPLAY(@MARJOLAINE)
8020 END MAIN
```

ANALYSE ET PROGRAMMATION

DANS UN SYSTEME EXPERT

Les systèmes experts sont des programmes classés dans la catégorie de l'"Intelligence Artificielle", terme obsequieux vis-à-vis d'une machine et d'un logiciel qui n'a d'intelligent que celle qu'on veut bien lui transmettre. Il serait plus juste de parler ici de "Raisonnement Artificiel" puisque ces logiciels, en general suivent une logique de raisonnement brute.

QU' EST-CE QU'UN SYSTEME EXPERT?

Maintes fois decrits dans ces pages (voir les articles de MM. LANG, TOUAJ et RAFORTHO, ainsi que l'article sur EXPERT-2 du groupe INTELLIGENCE ARTIFICIELLE, JEDI No 13), un système expert est une suite de règles ne suivant pas un algorithme défini, ces règles proposées à l'utilisateur se combinent pour tenter de parvenir à une hypothèse finale.

LA REGLE est une suite d'assertions qui aboutissent soit à une ou des conclusions, soit à une hypothèse finale.

L'ASSERTION est une affirmation primitive qui, soumise à l'utilisateur, peut prendre une des valeurs VRAI ou FAUX.

LA CONCLUSION est une affirmation globale qui prend la valeur VRAI si toutes les assertions précédentes sont déclarées VRAI, sinon elle prend la valeur FAUX. Cette conclusion, lorsqu' elle est prise comme assertion dans une règle qui suit, prendra la valeur V ou F prédéterminée précédemment.

L'HYPOTHESE est la conclusion finale à laquelle le système expert aboutirait si toutes les assertions de cette dernière règle ont été données comme V.

QU'EST-CE QU' UNE REGLE?

Une règle est un système logique d'assertions et de conclusions, comme nous l'avons dit plus haut. Ces assertions et ces conclusions sont liées entre elles par des opérateurs logiques. Ces opérateurs sont:

SI : forme interrogative dont la valeur V ou F n'est pas prédéfinie.

OU , ET : sans commentaires.

ALORS : relation d'implication.

NON : négation d'une assertion ou d'une conclusion.

Prenons un petit exemple de programme:
"Si la terre est sèche et que je n'arrose pas, les plantes vont flétrir. Mon jardin ne sera pas beau".

Nous avons dans cet exemple:
- 2 assertions: La terre est sèche
  J'arrose
- 1 conclusion: Les plantes vont flétrir
- 1 hypothèse : Mon jardin ne sera pas beau
- 4 opérateurs logiques: SI, ET, ALORS, NON.

Ecrivons directement les règles qui s'en dégagent:

```
(Règle 1)
SI la terre est sèche
ET SI NON j'arrose
ALORS les plantes vont flétrir

(Règle 2)
SI les plantes vont flétrir
ALORS HYPOTHESE mon jardin ne sera pas beau

(Règle 3)
SI NON les plantes vont flétrir
ALORS HYPOTHESE mon jardin sera beau
```

Deux points sont à noter dans ce petit programme. D'abord l'hypothèse de la règle 3 se dégage implicitement de la phrase de départ, et qu'elle est vrai dans tous les cas. (Sauf si à force d'arroser, mes plantes pourrissent, d'accord..). D'autre part il est bien évident que si une règle ne prouve pas l'hypothèse, elle ne prouve pas non plus son contraire. Il ne faut pas oublier que l'opération ALORS est la relation "implique" (=>). La table de vérité de cette relation est la suivante:

A	B	A=>B
1	1	1
1	0	0
0	1	?
0	0	?

C'est pourquoi la règle 3 a été ajoutée expressément. Si la règle 3 n' existait pas, le système aurait répondu "Je ne sais pas conclure" ou quelque chose d'approchant.

COMMENT ANALYSER UN PROBLEME?

L'analyse, comme en programmation algorithmique normale, est essentielle: d'abord pour la rapidité de recherche dans le système, ensuite pour la compréhension des phases de recherche pour l'utilisateur. Si l'on commence par taper un programme directement "pour voir si ça marche" on est certain d'aboutir à des impasses ou à des impossibilités. Il est donc impératif de se donner quelques règles simples pour commencer à programmer.

->1) Partir des HYPOTHESES. TOUTES les HYPOTHESES.

->2) Faire un tableau général de toutes les ASSERTIONS aboutissant à chaque hypothese

->3) Sérier les assertions en GROUPES GENERAUX.

->4) Diviser équitablement en PARTIES LOGIQUES les hypothèses par rapport aux groupes

->5) Former chaque REGLE en COMMENTANT chaque fois que possible.

CAS CONCRET: Pour concrétiser ce cheminement de pensées, essayons d'effectuer un expert en pathologie du batiment concernant, plus précisément l'humidité dans les constructions (chacun sa spécialité). Nous utiliserons, pour celà, un système expert implanté en FORTH, assez rustique mais suffisant dans ce cas: EXPERT 2 (Copyright J. PARK & MOUNTAIN VIEW PRESS).

->1) Partir des HYPOTHESES, c'est-à-dire non pas des causes d'humidité mais les remèdes à apporter. Faisons la liste des causes:

1 Nappe phréatique
2 Eau de ruissellement à terre
3 Pluie battante
4 Condensation intérieure
5 Canalisation intérieure qui fuit
6 Toiture
7 Goutières

Toute forme d'accident est exclue, bien entendu. Ces cas sont plutot de la compétence d'un expert...d'assurances.

Cette liste va nous fournir les hypothèses de départ:

1 Drain périphérique
2 Coupure hydraulique des maçonneries
3 Réfection des façades extérieures
4 Isolation thermique et ventilation des locaux
5 Réfection des installations de plomberie ou de chauffage
6 Réfection de la toiture
7 Nettoyage ou réfection des chéneaux et descentes pluviales

→

->2) Faire un TABLEAU DES ASSERTIONS, regroupant les conséquences possibles de chaque cause, et noter là ou les hypothèses concernées. Le tableau 1 reprend la majeur partie des conséquences.

->3) Sérier en GROUPES GENERAUX les assertions par affinités ou par localisation (comme dans cet exemple) afin d'en tirer des conclusions générales qui auront pour but de dissocier le programme et ainsi de permettre la recherche rapide des hypothèses.
Le tableau 2 donne un exemple de groupement par localisation dans le batiment.

->4) DIVISER en partie logiques LES HYPOTHESES en partant du tableau 2.
Bien sur, il eut été possible d'effectuer directement le programme à partir du tableau 1, mais la complexité des règles ne pourrait fournir une rapidité de recherche ni au logiciel ni à l'utilisateur.
L'analyse du tableau nous révèle que pour la localisation 1, les hypothèses H1 et H2 sont prises en compte si cette localisation est vraie, et H3 à H7 si elle est fausse.
La localisation 3 partage les hypothèses de cette manière: H1, H4, H5 et H6 sont prises en compte si V, H2, H3, H7 si F.
Les hypothèses sont donc partagées en deux parties équitables à partir de la localisation 3 et l'arbre de recherches en sera réduit d'autant.

->5) Former chaque REGLE en COMMENTANT si possible: nous n'avons pas parlé des commentaires. Il est bien évident que dans un programme source classique, le commentaire aide non seulement le programmeur à retrouver ou à se souvenir de chaque partie de son programme mais, de plus les lecteurs dudit programme auront une idée du principe et du fonctionnement de sa structure.
Dans le logiciel employé (EXPERT-2), les commentaires ont une utilité supplémentaire: à l'appel de l'utilisateur lors de la proposition d'une assertion, trois possibilité de choix sont offertes:

-OUI, cette assertion est vraie
-NON, cette assertion est fausse
-POURQUOI cette règle est-elle soumise à l'utilisateur

et ce pourquoi renvoie à l'écran un PARCE QUE inclus dans la règle qui sert à expliciter les assertions employées. De plus toutes les assertions, les conclusions et les hypothèses de la règle sont transmises à l'écran à la suite d'un commentaire du type: "Je tente de prouver..."
Il est donc utile, sinon indispensable d'expliciter (succintement mais efficacement) une règle.
Le choix du commentaire ne doit pas se borner à l'action que le logiciel tente d'effectuer, mais bien à l'explication du choix des assertions et des hypothèses, à leur cause et à leur conséquence, en laissant de coté l'outil pour ne se consacrer uniquement qu'au sujet traité.
C'est pourquoi, lorsque l'on part des hypothèses, la première liste des causes d'humidité devrait servir à alimenter les commentaires sur les asertions concernées.
Pour se résumer: les causes seront les commentaires, les conséquences seront les assertions et les remèdes seront les hypothèses.
Passons à la programmation proprement dite.

```
( REGLE 0 )
SI Humidité à l'intérieur du batiment
ALORS Eaux intérieures
```

Cette conclusion regroupe les hypothèses H1, H4, H5 et H6.
La règle contraire:

```
( REGLE 1 )
SINON Humidité à l'intérieur du batiment
ALORS Eaux exterieures
```

Celle-ci regroupe les hypothèses H2, H3, H7.
Nous voilà en présence de deux groupes de phénomènes distincts. En fait ces deux règles sont superfétatoires mais renforcent bien les conclusions pour la suite des règles.
Une règle supplémentaire découle immédiatement du tableau 2 et conduit à une hypothèse directe: les localisations 1 et 3 amènent H1.

```
( REGLE 2 )
SI Eaux intérieures
ET Humidité dans partie la plus basse du batiment
PARCEQUE L'humidité vient de la nappe phréatique par capillarité
ALORSACTION DRAIN
```

Cette forme de conclusion propre à EXPERT-2 sera développée ultérieurement.
La règle suivante parait curieuse au premier abord, mais en fait, les assertions qui la compose ne sont pas contradictoires. Cette règle délimite les deux hypothèses H1 et H4 et s'énnonce:
Si l'humidité est à l'intérieur du batiment sauf sur les façades,
Et si l'humidité est situé sur la façade
Alors...l'humidité est partout!
L'assertion qui suivra prendra en compte la localisation du batiment et déterminera le choix sur l'une des hypothèses H1 ou H4.

```
( REGLE 3 )
SI Eaux intérieures
ET Humidité sur façades (intérieur ou extérieur)
ALORS Eaux réparties
```

Et, en combinaison avec la règle 2, nous obtenons l'hypothèse H4.

```
( REGLE 4 )
SI Eaux réparties
ETNON humidité dans la partie la plus basse du batiment
PARCEQUE L'humidité provient d'une mauvaise conception intérieure
ALORSACTION ISOL-VENTIL
```

A ce stade de la programmation nous avons:

```
Local. 3 > eaux intérieures > H1-H4-H5-H6
Local. 2 > eaux exterieures > H2-H3-H7
Local.2 et 3 > eaux réparties > H1-H4
             > DRAIN       > H1
             > ISOL-VENTIL > H4
```

Revenons à l'opérateur-mystère:

QU'EST-CE QUE "ALORSACTION" ?

La puissance du logiciel EXPERT-2 est toute contenue dans les super-opérateurs finissant par -ACTION.
SIACTION SINONACTION ETSIACTION ETNONACTION ALORSACTION ETALORSACTION PARCEQUEACTION
Ces opérateurs sont des appels à des sous-programmes FORTH placés avant les règles et qui permettent, en les plaçant dans les assertions, de prévoir autant de calculs, tests, moyennes, comparaisons et commentaires que le programmeur le désire.
La seule contrainte à respecter, à part l'obligation de ne pas "planter" le programme, est de laisser sur la pile de données un indicateur V(rai) (n'importe quel nombre sauf 0) ou F(aux) (0) à la fin du sous-programme. Cet indication peut résulter de tests effectués à l'intérieur du sous-programme ou bien de l'action voulue du programmeur, comme nous allons le voir à la suite du programme.

→

Apres cette disgression, retournons à nos règles et occupons-nous maintenant du tableau 1. Les assertions qui s'en dégagent ne sont pas liées les unes aux autres. Il nous aurait fallu un opérateur d'union "OU" pour les prendre en compte dans leur globalité. Malheureusement, le concepteur du logiciel EXPERT-2 ne l'a pas prévu. Il faudra donc dégager des règles communes en tenant compte du tableau 2 des localisations.

(REGLE 5)
SI Sol froid et humide
ET Eaux intérieures
ET Partie basse du batiment
PARCEQUE l'humidité de la nappe phréatique remonte dans le sol
ALORSACTION DRAIN

(REGLE 6)
SI Intérieur façade humide
ETNON Partie basse du batiment
ET Humidité localisée
PARCEQUE L'eau de pluie s'infiltre dans la façade
ALORSACTION FACADE

(REGLE 7)
SINON Eaux reparties
ET Interieur des facades humides
ALORSACTION COUPURE

Et ainsi de suite jusqu'à épiusement des assertions.

(REGLE 8)
SINON Eaux réparties
ET Revetement mural dégradé
ALORSACTION COUPURE

(REGLE 9)
SI Eaux réparties
ET Revetement mural dégradé
ALORSACTION DRAIN

(REGLE 10)
SI Humidite en partie haute du batiment
ET Interieur des façades humides
ALORSACTION CHENEAUX

....

Et ainsi de suite jusqu'à la fin des assertions, à part que cela ne marche pas très bien puisqu'il faut placer la partie ... ACTION dans la règle qui contient l'hypothèse, comme une assertion qui aurait sa propre réponse (cette manière de faire étant très particulière d'EXPERT 2, nous parlerons de la programmation des autres systèmes, comme FORTHLOG, lors d'un prochain numéro).

Donc il aurait fallu appeler une conclusion à la place des sous-programmes (voir le listing du programme Humidité), cette conclusion étant de la forme:

SI (appel)
ETACTION (partie action)
ALORSHYP (hypothèse)

la partie action reprenant les explications necessaires à l'expertise considérée, elle est automatiquement générée par la conclusion SI (appel) et se comporte comme une assertion qui contiendrait sa propre valeur VRAI et qui concluerait par l'impression de l'hypothèse à sa suite.

Un mot encore pour bien comprendre la puissance des opérateurs ...ACTION. Ces mots pourraient parfaitement "forgetter" une application en cours et la remplacer par une autre, en controlant l'appel de la pile de retours. Cela veut dire qu'une application comme ce système d'expertise d'humidité peut très bien faire partie intégrante d'un système général de pathologie de batiment, avcec ses différentes applications concernant le gros oeuvre (fissurations, fondations etc.), les économies d'énergie (calculs de coefficients de déperditions calorifiques avec les solutions générées automatiquement par le système), la maintenance des équipements (chaudières, ascenseurs etc.) et l'exploitation en général. Ces systèmes étant chapeautés par un métasystème servant d'aiguillage et guidant l'utilisateur vers ses problèmes particuliers.

Une autre possibilité offerte par ces opérateurs est le controle de bus recevant des informations extérieures (possibilité d'écrire toute l'exploitation d'une gestion technique centralisée en FORTH et de manipuler les données reçues et transmises par le système expert).

EN GUISE DE CONCLUSION

S'il peut y avoir une conclusion à cet article, c'est bien celle de dire qu'il n'y aura pas de conclusion de sitot pour les systèmes experts, aussi rudimentaires soient-ils. Il se prépare de beaux jours pour ceux qui sont intéressés par la manipulation de ces logiciels qui n'ont de loin pas dit leur dernier mot.

Avec FORTH JEDI pour AMSTRAD-SCHNEIDER

```
SCR # 22
( EXPERT BATIMENT HUMIDITE                        23/10/85 )
: MUR ;
: COUPURE INVERSE CLS
." Les eaux de pluie atteignent le bas des facades situes"
CR ." a l'ouest et s'infiltrent par les microfissures."
CR ." Les remedes aux desordres causes par ces eaux sont d'"
CR ." un cout eleve: soit de couper les facades par parties"
CR ." et d'y mettre en place une feuille de plomb ou de"
CR ." bitume, soit de percer les murs pour y placer des"
CR ." poteries poreuses sur toute la peripherie, soit par"
CR ." 'electro-osmose', procede ionique empechant la migration"
CR ." des eaux." 1 CR INVERSE ;



;S
```

→

```
SCR # 24
( EXPERT BATIMENT HUMIDITE                     23/10/85 )
: FUITE INVERSE CLS
." L'humidite ne peut provenir que d'une fuite de canalisation"
CR ." a l'interieur du batiment. Cela peut provenir aussi bien"
CR ." du reseau d'eau que de celui du chauffage."
CR ." Les fuites les moins decelables sont celles ou les"
CR ." canalisations sont noyes dans les murs."
1 CR INVERSE ;
;S

SCR # 25
( EXPERT BATIMENT HUMIDITE                     23/10/85 )
 : DRAIN INVERSE CLS
 ." La nappe phreatique impregne les fondations"
CR ." et l'eau remonte par capillarite (par les microfissures"
CR ." des murs et des sols). Cette remontee peut atteindre"
CR ." jusqu'a 30 metres de hauteur dans de 'bonnes' conditions."
CR ." Une facon d'eviter cela est de placer un drain sous"
CR ." le niveau des fondations et ainsi tarir localement la"
CR ." nappe autour du batiment." 1 CR INVERSE ;
;S

SCR # 26
( EXPERT BATIMENT HUMIDITE                     23/10/85 )
: ISOL-VENTIL INVERSE CLS
." Le manque d'isolation en periode froide fait que" CR
." la difference de temperature interieure-exterieure" CR
." condense la vapeur sur les murs a l'interieur." CR
." Le defaut de ventilation des locaux empeche la vapeur" CR
." d'eau de se degager des pieces et ainsi condense sur" CR
." les murs, en particulier les cuisines et les salles d'" CR
." eau. Cette mauvaise conception interieure se remedie par:" CR
 ."    - L'isolation des combles (laine de verre ou roche)" CR
."    - Le doublage des parrois exterieures (panneaux" CR
."      sandwitchs)" CR
."    - L'extraction ou la ventilation dans les locaux " CR
."      humides" 1 CR INVERSE ;
;S

SCR # 27
( EXPERT BATIMENT HUMIDITE                     23/10/85 )
: CHENEAUX INVERSE CLS
." Les cheneaux engorges essentiellement par les feuilles" CR
." mortes en automne doivent etre netoyees au moins une" CR
." fois par an et les descentes pluviales verifiees tous" CR
." les cinq ans et changees des qu'elles sont en mauvais etat"
CR INVERSE 1 ;
;S

SCR # 28
( EXPERT BATIMENT HUMIDITE                     23/10/85 )
: TOITURE INVERSE CLS
." Il peut y avoir plusieurs causes de degradation de la" CR
." toiture: la vetusete, le vent, les nichees d' oiseaux," CR
." les racordements aux souches (solins, ruelees) degrades" CR
." ou une incompatibilite du type de couverture." CR
." La solution la plus raisonnable est de faire appel a" CR
." un homme de l'art et de demander une expertise et un" CR
." devis detaille."
CR INVERSE 1 ;
;S
```

→

```
SCR # 29
( EXPERT BATIMENT HUMIDITE                          23/10/85 )
: FACADE INVERSE CLS
." Les pluies battantes peuvent s'infiltrer dans les murs" CR
." par les fissures de la facade si l'enduit ou le crepi" CR
." est degrade. La solution est de revetir la facade d'un" CR
." enduit du type 'PLYOLITE' pour les supports fissures" CR
." et du type batard pour les supports en pierre calcaire." CR
." Les facades en pierre de meuliere ne peuvent etre" CR
." enduites par les produits existants a l'heure actuelle." CR
CR INVERSE 1 ;
;S

SCR # 30
( EXPERT BATIMENT HUMIDITE                          23/10/85 )
REGLES
( REGLE 0 )
SI HUMIDITE A L'INTERIEUR DU BATIMENT
ALORS EAUX INTERIEURES
( REGLE 1 )
SINON HUMIDITE A L'INTERIEUR DU BATIMENT
ALORS EAUX EXTERIEURES
( REGLE 2 )
SI EAUX INTERIEURES
ET HUMIDITE EN PARTIE LA PLUS BASSE DU BATIMENT
PARCEQUE L'HUMIDITE VIENT DE LA NAPPE PHREATIQUE
ALORS H1
;S

SCR # 31
( EXPERT BATIMENT HUMIDITE                          23/10/85 )
( REGLE 3 )
SI EAUX INTERIEURES
ET HUMIDITE SUR FACADES (INT. ET EXT.)
ALORS EAUX REPARTIES
( REGLE 4 )
SI EAUX REPARTIES
ETNON HUMIDITE SUR FACADES (INT. ET EXT.)
PARCEQUE L'HUMIDITE VIENT D'UNE MAUVAISE CONCEPTION INTERIEURE
ALORS H4
;S

SCR # 32
( EXPERT BATIMENT HUMIDITE                          23/10/85 )
( REGLE 5 )
SI EAUX INTERIEURES
ET HUMIDITE EN PARTIE LA PLUS BASSE DU BATIMENT
ET SOL FROID ET HUMIDE
PARCEQUE L'HUMIDITE VIENT DE LA NAPPE PHREATIQUE
ALORS H1
( REGLE 6 )
SI INTERIEUR DES FACADES HUMIDES
ETNON HUMIDITE EN PARTIE LA PLUS BASSE DU BATIMENT
ET HUMIDITE LOCALISE
PARCEQUE L'EAU DE PLUIE S'INFILTRE DANS LA FACADE
ALORS H3
;S

SCR # 33
( EXPERT BATIMENT HUMIDITE                          23/10/85 )
( REGLE 7 )
SINON EAUX REPARTIES
ET INTERIEUR DES FACADES HUMIDES
PARCEQUE L'EAU DE RUISSELLEMENT EXTERIEURE REMONTE EN FACADE
ALORS H2
( REGLE 8 )
SINON EAUX REPARTIES
ET REVETEMENT MURAL DEGRADE
ALORS H2
( REGLE 9 )
SI EAUX REPARTIES
ET REVETEMENT MURAL DEGRADE
ALORS H1
;S
```

```
SCR # 34
( EXPERT BATIMENT HUMIDITE                    23/10/85 )
( REGLE 10 )
SI REVETEMENT MURAL DEGRADE
ET HUMIDITE EN PARTIE LA PLUS HAUTE DU BATIMENT
ET INTERIEUR DES FACADES HUMIDES
ALORS H7
( REGLE 11 )
SI REVETEMENT MURAL DEGRADE
ET HUMIDITE EN PARTIE LA PLUS HAUTE DU BATIMENT
ETNON INTERIEUR DES FACADES HUMIDES
ALORS H6
;S

SCR # 35
( EXPERT BATIMENT HUMIDITE                    23/10/85 )
( REGLE 12 )
SI EAUX INTERIEURES
ET SENSATION HUMIDE EN PIECE FROIDE
ALORS H1
( REGLE 13 )
SI EAUX EXTERIEURES
ET SENSATION HUMIDE EN PIECE FROIDE
ALORS H2
( REGLE 14 )
SI EAUX INTERIEURES
ET PLINTHES POURRIES
ALORS H1
;S

SCR # 36
( EXPERT BATIMENT HUMIDITE                    23/10/85 )
( REGLE 15 )
SI EAUX EXTERIEURES
ET PLINTHES POURRIES
ALORS H2
( REGLE 16 )
SI SENSATION HUMIDE EN PIECE CHAUDE
ALORS H4
( REGLE 17 )
SI EAUX REPARTIES
ETNON HUMIDITE EN PARTIE LA PLUS BASSE DU BATIMENT
ET REVETEMENT MURAL DEGRADE
ALORS H5
;S

SCR # 37
( EXPERT BATIMENT HUMIDITE                    23/10/85 )
( REGLE 18 )
SI EAUX REPARTIES
ET REVETEMENT DE SOL DEGRADE
ALORS H1
( REGLE 19 )
SI EAUX REPARTIES
ET EXTERIEUR BAS DES FACADES HUMIDES
ALORS H1
( REGLE 20 )
SINON EAUX REPARTIES
ET EXTERIEUR BAS DES FACADES HUMIDES
ALORS H2
;S

SCR # 38
( EXPERT BATIMENT HUMIDITE                    23/10/85 )
( REGLE 21 )
SI EXTERIEUR HAUT DES FACADES HUMIDES
ALORS H7
( REGLE 22 )
SI EXTERIEUR BAS DES FACADES HUMIDES
ET HUMIDITE LOCALISE
ALORS H3
( REGLE 23 )
SI EAUX REPARTIES
ET ODEUR DE MOISI DANS UNE PIECE
ALORS H1
;S
```

```
SCR # 39
( EXPERT BATIMENT HUMIDITE                               23/10/85 )
( REGLE 24 )
SI FISSURATIONS DES FACADES
ALORS H3

( REGLE 25 )
SI SENSATION FROIDE VENANT DES FACADES
ALORS H4

( REGLE 26 )
SI BUEE INSISTANTE SUR MURS ET FENETRES
ALORS H4

;S
SCR # 40
( EXPERT BATIMENT HUMIDITE                               23/10/85 )
( REGLE 27 )
SI FONDS D'ARMOIRES HUMIDES
ALORS H4

( REGLE 28 )
SI TACHES SUR COFFRES BOIS
ALORS H5

( REGLE 29 )
SI HUMIDITE EN PLAFOND PRES D'UNE CLOISON
ALORS H5

;S

SCR # 41
( EXPERT BATIMENT HUMIDITE                               23/10/85 )
( REGLE 30 )
SI TACHES EN PLAFOND
ALORS H6

( REGLE 31 )
SI TACHE(S) EN PLANCHER DES COMBLES
ALORS H6

( REGLE 32 )
SI HUMIDITE EN SOUS-FACE DE COUVERTURE
ALORS H6

;S

SCR # 42
( EXPERT BATIMENT HUMIDITE                               23/10/85 )
( REGLE 33 )
SI HUMIDITE AU COIN DE PLAFOND D'UNE PIECE
ALORS H7

( REGLE 34 )
SI GOUTTIERE DEBORDE
ALORS H7

;S
SCR # 43
( EXPERT BATIMENT HUMIDITE                               23/10/85 )
SI H1
ETACTION DRAIN
ALORSHYP DRAIN PERIPHERIQUE
SI H3
ETACTION FACADE
ALORSHYP REFECTION ENDUITS
SI H4
ETACTION ISOL-VENTIL
ALORSHYP ISOLATION VENTILATION
SI H6
ETACTION TOITURE
ALORSHYP REFECTION TOITURE
;S
```

→

```
SCR # 44
( EXPERT BATIMENT HUMIDITE                          23/10/85 )
SI H7
ETACTION CHENEAUX
ALORSHYP NETTOYAGE CHENEAUX
SI H2
ETACTION COUPURE
ALORSHYP COUPURE HYDRAULIQUE
SI H5
ETACTION FUITE
ALORSHYP REFECTION TUYAUTERIES
FAIT
;S
ok
```

MUMPS

par Yannick LE GRAS

XII LA COMMANDE DE BOUCLE

A) La commande FOR

Cette commande doit retenir votre attention, car c'est l'un des outils les plus utilisés lors de la programmation en MUMPS. Cette commande, comme toutes les autres, ne s'applique qu'à la ligne dans laquelle elle se trouve. Plus clairement, la commande FOR permet de répéter les instructions suivant l'instruction FOR, ceci un nombre de fois connu ou non. D'autre part, la commande FOR peut être utilisée avec ce qu'on appelle, dans d'autres langages, des variables discrètes. Afin de ne pas faire des kilomètres de prose, il est plus simple d'analyser les trois types d'utilisation de la commande FOR. La forme abrégée de la commande FOR est : F .

```
F NB=0:2:8 W !,NB
```

L'exécution de cette ligne produit l'affichage suivant :

```
0
2
4
6
8
```

En clair, on demande à la machine de maintenir un compteur ordinal commencant à la valeur 0, puis s'incrémentant avec un pas de 2 jusqu'à la valeur maximale 8, et à chaque passage, d'écrire sur l'écran à la ligne suivante la valeur de NB.

```
F NB=1:2 Q:NB>8  W !,NB
```

Le résultat produit est :

```
1
3
5
7
```

Cette fois-ci, nous avons demandé une boucle sans fin avec un incrément de la variable NB à chaque passage égal a 2. L'arrêt de cette boucle est réalisé grâce à la commande Q (QUIT) postconditionnée par le test NB supérieur à 8.

```
F NB=10,"BONJOUR","MONSIEUR",20 W !,NB
```

Suite page 15

FORTH Chaînage de vocabulaire FIG-Forth

par Michel ZUPAN

Le FIG-Forth permet de structurer le dictionanire du langage en plusieurs vocabulaires selon un modèle arborescent qui semble parfois rigide une fois fixé. Il n'en est rien: il est parfaitement envisageable de ré-organiser un dictionnaire selon des besoins contradictoires tels que compilation et exécution.

Rappelons que deux vocabulaires sont accessibles: le vocabulaire courant où s'ajoutent les nouvelles définitions et le vocabulaire de contexte où s'effectuent les recherches, avec cette réserve qu'une définitin deux-points force le contexte au courant, tandis que DEFINITIONS fait simplement l'opération inverse.

Les chaînages sont réalisés par les champs paramètres des en-têtes de vocabulaires créés par VOCABULARY. Ces mots contiennent succesivement:

PFA	: renvoi au DOES> de VOCABULARY
PFA+2	: NFA 81AØ de mot fictif pointé par le LFA du premier mot d'un vocabulaire secondaire.
PFA+4	: LFA du mot fictif pointant le dernier mot du vocabulaire défini et éventuellement pointé par CONTEXT.
PFA+6	: chaînage VOC-LINK.

Le mot ALSO (---) va permettre de modifier ces liens de façon très simple.

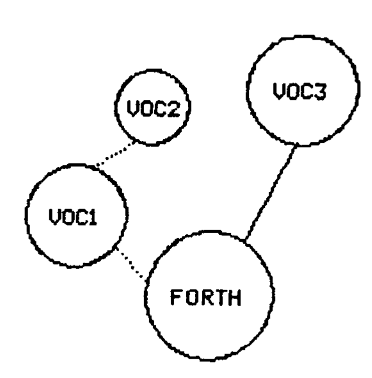
S'utilisant sous la forme

VOC2 ALSO VOC1

il réalise le chaînage de VOC1 à VOC2 avec un vocabulaire de contexte devenant la suite descendante

VOC2 - VOC1 - ... - FORTH

Appliquons ALSO à un exemple complexe. Supposons un dictionnaire ainsi organisé, tous vocabulaires immédiats:



Si nous devons définir dans VOC3 des mots faisant de fréquents appels aux mots de VOC2, il faudra répéter dans ces définitions des séquences

: ... VOC1 VOC2 ... VOC3 ... ;

L'appel de VOC1 avant VOC2 étant requis pour avoir accès au mot VOC2 lui-même défini dans VOC1.

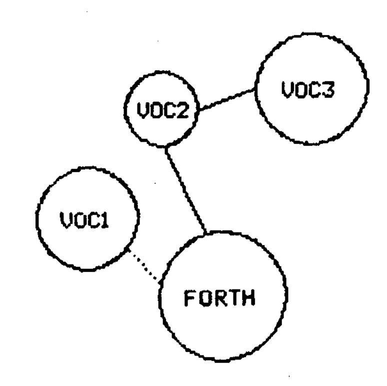
Précisons ici à cet égard qu'il est toujours préférable de définir un vocabulaire dans la racine FORTH, comme en F-79 (79-Standard), quitte à effectuer des chaînages ultérieurs.

Si nous écrivons

VOC3 ALSO VOC1 pour l'accès à VOC2 sous VOC3

VOC1 VOC2 ALSO FORTH
VOC3 ALSO VOC2
DEFINITIONS

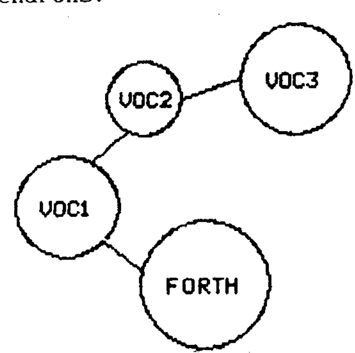
notre problème est résolu puisque nous disposons de la structure suivante:



Si nous écrivons encore

VOC1 VOC2 ALSO VOC1

nous obtiendrons:



Enfin VOC3 ALSO FORTH rétablira la configuration initiale, plus efficace à l'exécution.

La définition d' ALSO est simple dans son principe, mais comporte plusieurs sécurités:

- le mot objet de ALSO doit être un vocabulaire.
- nous conseillons une règle d'ordre pour interdire toute création d'un serpent se mordant la queue avec recherche infinie.
- le vocabulaire de contexte ne doit pas être vide.

Nous avons utilisé le quasi-standard ABORT" (flag ---) dont voici au besoin une définition:

```
: ABORT" ( flag --- )
  ?COMP
  [COMPILE] IF
  COMPILE CR
  [COMPILE] ."
  COMPILE ABORT ( ou COMPILE SP!
                     COMPILE QUIT )
  [COMPILE] THEN
; IMMEDIATE
```



```
SCRN# 75
( ALSO : FIG VOCABULARIES CHAINING          M.ZUPAN dec.85 )
FORTH DEFINITIONS HEX
: ALSO ( ALSO vocabulary ) ( --- )
       [COMPILE] ' DUP @ [ ' FORTH @ ] LITERAL =
        0= ABORT" NOT VOCABULARY ALSO ERROR "
       2+ CONTEXT @ 2DUP 2- <
        0= ABORT" EQUAL OR UPPER ALSO ERROR "
       @ DUP @ A081
         = ABORT" EMPTY CONTEXT ALSO ERROR"
       BEGIN DUP PFA LFA @ DUP @ A081 =
       0= WHILE SWAP DROP REPEAT
       DROP PFA LFA ! ; IMMEDIATE
DECIMAL ;S
```

Référence bibliographique :

FORTH par W.P. SALMAN, O. TISSERAND, B. TOULOUT
Editions EYROLLES 1983

Suite de la page 13

Dans ce cas la variable NB prendra, tour à tour, les valeurs suivantes :

```
10
BONJOUR
MONSIEUR
20
```

Note : Il est possible d'imbriquer plusieurs boucles FOR sur une même ligne.
L'incrément mentionné peut être négatif.

Attention ! A la rencontre d'un ordre QUIT dans une boucle FOR, l'interpréteur MUMPS abandonne la boucle courante.

MUMPS, de part sa puissance, permet de mixer les trois formes de FOR. Ainsi, il est possible d'écrire la ligne suivante :

```
F NB=5:5:25,10:-2:6,"BONJOUR",200,2:1:4 W NB," "
```

L'exécution produira à l'écran le résultat ci-dessous:

```
5 10 15 20 25 10 8 6 BONJOUR 200 2 3 4
```

Autre ligne, autre résultat !

```
S VAR="" F NB=1:1:3 S VAR=VAR_"A" F J=1:1:NB S VAR=VAR_"B"
```

Apres l'exécution de cette ligne la variable VAR contient: ABABBABBB

B) GOTO ET QUIT DANS LES BOUCLES FOR

Nous vous rappelons que l'interpréteur MUMPS ne travaille que sur des lignes de commandes. Autrement dit, à la rencontre d'un ordre GOTO, la ligne courante est abandonnée, l'exécution est reprise à l'etiquette ou à la routine specifiée. Si cet ordre apparait dans une boucle FOR, celle-ci est interrompue. Nos vous rappelons que l'ordre QUIT, dans une boucle FOR, stoppe celle-ci. Par conséquent, si un QUIT est mentionné dans une ligne ne comportant qu'une seule boucle FOR, l'exécution continuera à la ligne suivante.

Reprenons le dernier exemple du paragraphe précédent en y insérant un ordre QUIT :

```
S VAR="" F NB=1:1:3 S VAR=VAR_A F J=1:1 Q:J>NB  S VAR=VAR_"B"
W !,VAR
```

→

Nous allons détailler l'exécution de ces deux lignes de commandes :

- Attribution de la chaîne vide à la variable VAR
- Exécution d'une boucle permettant trois passages
- Concaténation de l'ancien contenu de la variable VAR avec le caractère A
- Exécution d'une boucle sans fin
- Abandon de la boucle sans fin lorsque J est supérieur a NB
- Concaténation de l'ancien contenu de la variable VAR avec le caractère B
- Affichage sur l'écran, à la ligne suivante, du contenu de la variable VAR

Rappel ! La commande QUIT n'ayant pas d'argument, elle doit être obligatoirement séparée du verbe qui la suit par deux espaces.

FORTHLOG Compléments

LES PREMIERS PAS

10 LOAD pour charger les mots tesl que CASE etc...
20 LOAD
41 LOAD pour charger FORTHLOG

Dans un premier temps, il faut construire sa base de connaissances sur deux écrans consécutifs, et ensuite sa base de faits initiaux sur un écran. Pour finir, taper le numéro du premier écran de la base de connaissances, le numéro de l'écran de la base de faits initiaux et en-fin le mot "MOTEUR"... Bonne chance.

LES EXEMPLES

Exemple 1: si F1 & X3 QLORS celà entraine X4 etc... Cet exemple propose un essai presque simpliste de calcul des propositions.

Exemple 2: cet exemple du domaine de la botanique illustre un autre cas de calcul des propositions. Il est tiré de l'ouveage "Introduction aux systèmes experts" de Mr GONDRAN.

Exemple 3: Cet exemple issu du même ouvrage, illustre un cas appliqué à la logique du premier ordre.

Exemple 4: cet exemple présente une sélection de quelques règles liées à un problème de diagnostic. La base de connaissance n'est que la traduction en FORTHLOG, d'une petite partie de l'organigramme du diagnostic, 5 mm après injection de sécurité dans une centrale PWR 900MW, cf. procédure A0 notes Moroni (1982) et Brillon, Janin, Munier (1981).

Exemple 5: voici un exemple qui ne relève pas spécialement du système expert, le calcul d'un bulletin de paie. Pourtant celà fonctionne et vous donne un net à payer !

CONCLUSION

Ce langage qui tient en 7 K de mémoire et qui solutionne des problèmes -bien sûr simples - avec un temps de réponse plus qu'honorable (merci FORTH!) peut être perfectionné et enrichi par toute personne connaissant FORTH. En effet, en fonction de l'application, chacun pourra très facilement intégrer de nouveaux opérateurs, et ainsi enrichir personnellement FORTHLOG (les écrans 34 à 41 restent disponibles). Alors à vos micros et bonne chance.

```
SCR # 10
( E10 DEFINITION STRING SIMPLIFIE )
: STRING      ( N --- )
   <BUILDS
     0 ,          ( PFA +2)
     ALLOT        ( PFA +4)
   DOES>
;



: $S            ( --- )
   38 WORD HERE COUNT
   1+  ( 1+ DANS NBRE OCTET A DEPLAC.)
   SWAP
   1-  ( -1 DANS ADDR. DEBUT ZONE A
       MOUVEMENTER )
   ROT ROT CMOVE
;

-->
```

```
SCR # 11
( E11 COMPARAISON CHAINE CARACT. -TEXT )
: -TEXT     ( ADR1 U ADR2 --- FLAG)
   OVER OVER + SWAP
   DO
     DROP 2+ DUP 2- @ I @ - DUP
     IF
       DUP ABS / LEAVE
     THEN
     2
   +LOOP
   SWAP DROP
```

```
SCR # 12
( E12 INSTRUCTION CASE )

: CASE
   ?COMP CSP @ !CSP 6 ; IMMEDIATE

: OF
   6 ?PAIRS COMPILE OVER COMPILE =
   COMPILE 0BRANCH HERE 0 , COMPILE
   DROP 7 ; IMMEDIATE

: ENDOF
   7 ?PAIRS COMPILE BRANCH HERE 0 ,
   SWAP 2 [COMPILE] ENDIF 6 ; IMMEDIATE

: ENDCASE 6 ?PAIRS COMPILE DROP
   BEGIN
     SP@ CSP @ = 0=
   WHILE
     2 [COMPILE] ENDIF
   REPEAT
   CSP ! ; IMMEDIATE
```

Documentation manquante
pour le Forth 83 NVSS

Tom ALMY

Ce document explique comment utiliser certaines des options dont l'usage n'est pas expliqué dans la documentation du F83 accompagnant les supports logiciels (Laxen et Perry) de FORTH.

OPERATIONS ELEMENTAIRES
SUR LES FICHIERS DU SYSTEME

Plusieurs fichiers peuvent être ouverts en une fois. Une fois un fichier ouvert, il n'est jamais refermé. (ce qui cause de gros problèmes avec CP/M -- j'y travaille en ce moment). Chaque tampon de block a un pointeur vers le FCB. La variable utilisateur FILE pointe sur le FCB courant.

FILE? -- affiche le nom du fichier courant.

CAPACITY -- retourne le numéro de bloc le plus élevé dans un fichier, ou zéro s'il n'y a pas de fichier.

DEFAULT -- ouvre le fichier dans la ligne de commande utilisant FCB appelée FCB1. Aucune extension de fichier par défaut n'est utilisée.

Pour ouvrir un nouveau fichier, nous devons en créer un neuf, par initialisation de fcb. Le vocabulaire CP/M a un FCB approprié nommé FCB2.

!FCB nom FCB-ADR --- Met le nom dans FCB à l'adresse FCB-ADR.

OPEN-FILE FCB-ADR --- Ouvre le fichier mentionné s'il n'est déjà ouvert. Utilisé après !FCB, et avant le stockage de l'adresse FCB dans la variable FILE.

Les mots suivants sont très utiles pour une maintenance de fichier particulère.

DEFINE --- filename Crée une entrée dans le dictionnaire nommée 'filename' pour des fichiers de même nom. Le fichier est ouvert, par conséquent, il doit exister au préalable. L'exécution du nom du fichier provoque son ouverture (si ce n'est déjà fait) et le rend fichier courant. Si cette entrée dans le dictionnaire existe déjà, alors cette commande sera sans effet.

OPEN --- filename Identique à DEFINE, mais le fichier devient courant immédiatement.

Les mots suivants sont relatifs à la gestion de fichier du système hôte:

DIR --- Affiche la directory du disque courant.

CREATE-FILE blocs --- filename Crée un nouveau fichier d'écrans de taille nblocs. Les écrans sont initialisés avec des espaces.

MORE nblocs --- Rajoute le nombre de blocs indiqué au fichier courant. Malheureusement, il fonctionne mal, car le fichier n'est JAMAIS fermé. Si vous fermez le fichier (par un appel système approprié) alors il fonctionnera correctement.

SAVE adr long --- filename Sauvegarde la mémoire sur le disque. Pour d'autres mots cette commande crée un fichier binaire exécutable.

Les commandes suivantes sont utilisées pour charger des écrans depuis un autre fichier que le fichier courant. Voir la section Copie d'écran dans ce document pour les commandes multi-fichiers.

FROM --- filename Réalise un DEFINE sur filename. Affecte le pointeur fcb, >FROM, dans le vocabulaire FILE vers ce fcb. Change le contexte de FILES.

LOAD --- écran mot du vocabulaire FILES La version de LOAD charge depuis le fichier spécifié dans la variable >FROM.

Exemple: FROM FOO.BLK 10 LOAD charge l'écran 10 du fichier FOO/BLK. Le fichier courant est modifié pendant l'exécution de LOAD. Une erreur ne restaure pas le fichier initial.

SUPPORT D'ECRANS COMMENTAIRES

NDT: littéralement 'Shadow Screen Support'. On traduira par écrans commentaires.

Lors de l'utilisation d'écrans commentaires, le fichier d'écran est effectivement divisé en deux parties. Le code forth est placé dans un écran, et son écran commentaire reçoit la documentation afférente. La différence entre les numéros d'écrans source et les numéros d'écrans commentaires est déterminée par DISPLACEMENT (un mot du vocabulaire SHADOW), dont le contenu est égal à CAPACITY 2/. Ceci signifie que le déplacement diffère en fonction de la taille du fichier.

A --- Change la valeur de SCR en son équivalent écran commentaire.

Voir les sections Copie d'écran, Edition et Listage, pour plus d'information concernant les mots du support écrans commentaires.

COPIE D'ECRAN

COPY from to --- Copie d'un écran simple vers un fichier simple. La copie d'écrans multiples est réalisée par d'autres moyens (il est plus aisé de décrire la technique que de définir le mot, et la technique ne dépend pas nécessairement de la définition du mot!).

HOP et CONVEY n HOP m 1 CONVEY Déplacement de blocks m à 1 vers les blocs m+n à 1+n.

m 1 TO n CONVEY Déplacement des blocks m à 1 depuis les blocs commençant à n. n doit être un nombre entier car il est traité par TO!

Pour déplacer des écrans entre des fichiers, utiliser le mot FROM pour spécifier le fichier d'origine (et changer le vocabulaire de contexte en FILES) puis la version FILES de COPY et CONVEY est utilisée. Toute erreur ne restaure pas le fichier courant. Exemple:

FROM FOO.BLK 2 3 TO 10 CONVEY FORTH

déplace l'écran 2 du fichier FOO.BLK vers l'écran 10 du fichier courant, et l'écran 3 du fichier FOO.BLK vers l'écran 11 du fichier courant.

. Dans le vocabulaire SHADOW, CA copiera l'écran courant (celui spécifié par la variable SCR) en son écran écran commentaire. Ainsi, les mots COPY et CONVEY copieront les écrans et leurs écrans commentaires associés dans un fichier. Les mots copiant les écrans et leurs écrans commentaires sont laissés à titre d'exercice au lecteur.

LISTAGE ET AFFICHAGE

VOCS --- Liste tous les vocabulaires. Les vocabulaires sont chainés manière identique au FIG-Forth.

ORDER --- Affiche l'ordre de recherche (schéma des vocabulaires dans ALSO ONLY).

→

WORDS --- Liste les mots du vocabulaire de contexte. Les variables LMARGIN et RMARGIN spécifient les marges gauche et droite de ce listage.

nées.

L --- Liste les écrans (spécifiés par la variable SCR).

LIST n --- Met l'argument spécifié dans SCR, puis liste l'écran.

N --- Ecran suivant (incrémente SCR).

B --- Ecran précédent (décrémente SCR). Voir aussi "A" dans la section ECRAN COMMENTAIRE.

TRIAD --- Oubliez l'existence de ce mot.

INDEX n m --- Liste la première ligne des écrans n à m.

IND n --- Idem INDEX, mais depuis l'écran n jusqu'à la fin du fichier.

Taper sur Control-P pour activer et désactiver l'imprimante.

Un jeu de définitions spécial permet l'impression sur une imprimante 132 colonnes. Dans ce cas, les écrans sont imprimés simultanément par paires. Les écrans vides ne sont jamais imprimés. L'écran zéro (valeur de la constante LOGO) est utilisé pour remplir la dernière page.

INIT-PR --- Mot vectorisé sélectionnant la fonction d'impression en 132 colonnes (si nécessaire). Sélectionner EPSON pour une imprimante Epson MX-80 ou équivalent, dans les autres cas, choisissez vos propres commandes.

Il sera nécessaire de recompiler le code pour modifier les en-têtes/terminaisons.

SHOW first last --- Imprime les écrans dans l'intervalle firts à last.

SHOW dans le vocabulaire SHADOW imprime les écrans et leurs écrans commentaires associés les uns à la suite des autres.

LISTING --- Imprime un listing des écrans commentaires de la totalité du fichier, ceci depuis l'écran 1.

DUMP adr len --- Réalise un dump ASCII et hexadécimal. La première ligne indique la position de départ du premier octet de chaque ligne, ceci pour un affichage de seize octets par ligne.

DU adr --- Réalise un dump sur 64 octets à partir de adr. Laisse sur la pile adr+64 pour une prochaine exécution de DU.

DL NoLigne --- Dumpe la ligne NoLigne de l'écran courant (SCR).

SUPPORT MULTITACHE

Le FORTH NVVS a un support multi-tache rudimentaire. Chaque tache a sa propre pile de données, "HERE" et "PAD", mais reste insuffisant pour des opérations multi-utilisateurs. Les fonctions multi-tâches ne sont actives seulement lors du passage du contrôle explicite ou lors d'opérations d'entrées/sorties (KEY EMIT BLOCK ou dérivées). Les mots suivants permettent le contrôle et la création de tâches:

MULTI --- Exécution en mode multi-tâches. Exécuter ce mot avant de créer une nouvelle tâche.

SINGLE --- Arrête le mode multi-tâche.

PAUSE --- Renvoie le contrôle; permet l'exécution d'une autre tâche. Ce mot est défini dans le système des primitives d'entrées/sorties.

>TYPE --- Version multi-tâche de TYPE. A utiliser quand le programme source est un block issu d'un disque Forth.

LOCAL tâche adr --- adr Renvoie une tâche et une variable utilisateur, retourne les adresses des variables utilisateurs de la tâche courante.

SLEEP tâche --- Met en sommeil la tâche spécifiée. Elle ne sera plus exécutée ultérieurement.

WAKE tâche --- Réveille la tâche indiquée. Si elle était endormie, son exécution démarre.

STOP ---- Demande à la tâche courante de se mettre en sommeil. Généralement placé à la fin du code d'une tâche.

TASK: taille --- Définit un mot pour de nouvelles tâches. La pile de retour de la tâche est de 100 octets, idem pour la pile de données.

SET-TASK: codebody taskname --- Initialise la tâche donnée pour exécution du code spécifié. Voir exemple:

Les tâches peuvent être initialisées de trois manières:

```
: FOO taskname ACTIVATE task code ;
```

exécute "task code" en tant que tâche annexe utilisant la tâches prédéfinie task.

```
BACKGROUND taskname task code ; taskname WAKE
```

crée une tâche annexe, taskname, et renvoie l'exécution sur le code de la tâche annexe.

```
tasksize TASK: taskname
: code task code ;
' code taskname SET-TASK
taskname WAKE
```

est un moyen manuel de créer une tâche.

L'EDITEUR

C'est tout simplement une abomination, mais en fait, s'adapte aux affichages simplifiés. Il est basé sur l'éditeur de "DEBUTEZ EN FORTH" (Starting Forth - en français aux éditions Eyrolles, NdT). Le système doit être configuré en fonction de votre terminal. Avec un peu de chance, "EDITOR nomduterminal FORTH" fera ce travail. Si le système ne connait pas votre terminal, vous devrez modifier le code (pas difficile ??) par adjonction des descriptions des fonctions de contrôle. Voir le programme source.

L'éditeur vous demande votre ID (identification) lors du premier appel. L'identificateur standard est jjmmmaaiii où jj est le jour, mmm est le mois, aa est l'année et iii sont vos initiales. Lors de la modification du contenu d'un écran, votre cachet est rajouté à droite de la première ligne. Généralement, cette ligne est réservée au commentaire et débute par le signe "barre de fraction inverse".

Je trouve cet éditeur innaproprié, difficile, sujet aux erreurs de manipulations, etc.. Mais ci-après se trouvent les commandes qui ont été rajoutées par rapport au livre DEBUTEZ EN FORTH.

EDIT n --- Edite l'écran n.

ED --- Réédite l'écran pointé par le contenu de SCR.

WHERE --- Mot vectorisé qui peut pointer sur EDIT pour une auto-édition en cas d'erreur. Voir la section MOTS VECTORISES.

QUIT --- Abandonne l'éditeur.

DONE --- Identique à QUIT, mais met à jour le cachet de marquage de l'heure.

→

C n --- Mouvement relatif de caractère dans une ligne.

+T n --- Mouvement relatif de ligne.

TOP --- Identique à "0 T".

sertion et de recherche.

KEEP --- Met la ligne courante dans le tampon d'insertion.

K --- Echange le contenu des buffers de recherche et d'insertion.

W --- Raccourci pour "SAVE-BUFFERS".

N --- Ecran suivant. Met à jour le cachet de marquage horaire.

B --- Ecran précédent. Met à jour le cachet de marquage horaire.

O --- Fonctionne comme I, mais surimprime le texte.

SPLIT --- Fait une césure de ligne à la position courante du curseur.

JOIN --- Fait une copie de la ligne suivante après la position courante du curseur.

G scr line --- prends line et l'insère au début de la ligne de scr après BRING. Prélève plusieurs lignes.

BRING scr first last --- Prélève plusieurs lignes.

JUST --- Identique à TILL mais n'inclut pas l'effacement de la chaîne.

KT --- Comme TILL, mais n'efface pas le texte.

NEW n --- Démarre la surimpression de lignes, à partir de la ligne n, ceci tant qu'une ligne vide n'est pas tapée.

SHADOW G --- Prend la ligne depuis un écran commentaire.

SHADOW BRING --- Prends des lignes depuis un écran commentaire.

FROM filename G --- Prends une ligne depuis le fichier filename.

FROM filename BRING --- Prends des lignes depuis le fichier filename.

MOTS VECTORISES

Utilisez ces commandes pour des mots vectorisés.

DEFER nom --- Définit un mot en tant que mot vectorisé.

IS ' mot IS nom permet l'exécution de "mot" lors de l'exécution du mot vectorisé.

IS est un mot immédiat, dépend donc de l'état. Dans une définition "deux-points" , utiliser ' entouré des crochets carrés.

LES VARIABLES UTILISATEUR

Les variables utilisateur sont allouées en tant que variables locales pour chaque tâche. Chaque tâche travaille avec sa copie de variables utilisateur.

Le nombre maximum de variables utilisateur est spécifié lors de la compilation. Aucun mot particulier ne peut être rajouté et employé. Les mots peuvent être rajoutés pour d'autres tâches.

Voici un bogue dans le code qui peut être corrigée comme suit:

```
' CREATE USER ' CREATE >BODY ! FORTH
```

#USER --- Une variable qui indique le nombre d'octets contenus dans la zone variable utilisateur.

USER --- Vocabulaire contenant les mots suivants:

DEFER --- Version en variable utilisateur du mot Forth DEFER.

VARIABLE --- Votre variable utilisateur commune.

CREATE --- Version utilisateur de CREATE. Utilisé par DEFER et VARIABLE.

ALLOT --- Alloue de l'espace dans la zone utilisateur.

LE DEBOGUEUR

Un seul mot peut être "débogué" (tracé) à la fois. Ce doit être un ,ot de type "deux-points" ou un ,ot vectorisant une définition "deux-points". N'essayez pas de déboguer d'autres types de mots. Il ne fonctionnera pas et le système se plantera problablement.

Pour déboguer un mot, exécuter "DEBUG mot". Lors de l'exécution de ce mot, le programme débogueur sera appelé automatiquement. Vous pouvez alors visualiser la définition pas à pas. Les commandes sont:

C - affichage continu (taper autre chose que C, F, ou Q pour en sortir).
F - entrée dans l'interpréteur FORTH. Exécuter RESUME pour continuer.
Q - abandonner le déboguage de ce mot.
toute autre touche - un pas.

FACILITE D'EXPLORATION

Vous pouvez explorer la définition de tout mot (à condition que le fichier source soit présent) par exécution de "VIEW mot". Vous pouvez explorer la définition de mots définis dans le fichier courant, mais la visualisation de la définition de mots appartenant à d'autres fichiers doit être réalisée en prenant certaines précautions:

1. Le fichier à "explorer" doit avoir une entrée dans le catalogue réalisée à l'aide des mots DEFINE ou OPEN.

2. "n VIEW= C!" pour spécifier le numéro du fichier courant à explorer. Il doit être compris dans l'intervalle 1 - 15, mais 1 à 4 sont utilsés par le système pour les fichiers source du système.

3. "n ' filename >BODY 40 + C!" associe le numéro de port avec le fichier, ainsi lors de la sélection du fichier, les nouvelles définitions ont un numéro de fichier correct.

4. " ' filename VIEW-FILES n 1- 2* + !" provoque l'association du fichier avec le numéro de port (nécessaire pour la commande d'exploration).

5. Maintenant, chargez les définitions depuis le fichier, et ensuite (probablement) sauvegardez le système sur le disque.

Il y a aussi la commande "SEE nom" pour décompiler les mots. Il ne requiert pas la présence du disque.

MOTS SUPPLEMENTAIRES DU NOYAU

Il y a dans le F83 des mots complémentaires au standard qui peuvent se révéler très utiles dans les programmes. Pour cause, l'utilisation de ces mots rend le programme non standard en regard du F83.



Opérateurs mathématiques:

2* 8* U2/ D> 0<= 0<= >= <= U>= U> U<= s'expliquent tout seuls.
0<> <> où "<>" signifie "non égal à".
WITHIN n min max --- min <= n < max
BEETWEN n min max --- min <= n <= max.
S>D n --- d du fig-Forth.
FLIP échange des octets d'un entier déposé sur la pile.
U*D identique à UM*.
MU/MOD du u --- r dq identique au fig M/MOD.
M/MOD dn n --- r q identique au fig M/.
D n n --- d identique au fig M.

Constantes et variables:

FALSE constante 0.
TRUE constante -1.

Compléments de manipulation de pile et de dictionaire:

P@ P! lit et écrit sur le port E/S.
-ROT n1 n2 n3 --- n3 n1 n2
NIP n1 n2 --- n2
TUCK n1 n2 --- n2 n1 n2
OFF identique à FALSE SWAP !
ON identique à TRUE SWAP !
CSET val adr --- logique ou valeur dans l'adresse.
CRESET comme CSET mais efface les bits sélectionnés.
CTOGGLE comme CSET mais active (XOR) les bits sélectionnés.

Traduction: MP le 24 mai 1986.
Référence: GOFIG GAZETTE; Greater Oregon Forth Interest Group.
Contact: Tom ALMY 692-2811 Tim HUANG 289-9135.
Adresse: GOFIG to Pann McCUAIG, 435 NW 27th St. Corvallis, OR 97330 USA
tel:(503) 752-5113

JEDI est une publication mensuelle éditée par JEDI, Association Loi de 1901. Cet exemplaire a été tiré à 500 exemplaires. Président : Michel ROUSSEAU
Secrétaire général : Marc Petremann.
Trésorier : Françoise Bolotin
Dépôt légal : Préfecture de Paris.
N° de Commission Paritaire : en cours

Il a été tiré cinq cent exemplaires de ce numéro.

Achevé d'imprimer sur les presses de l'imprimerie COPY-TOP

Imprimeur:
S.A. ASHBAY COMMUNICATION, 162 rue du Fg St. Honoré - 75008 PARIS